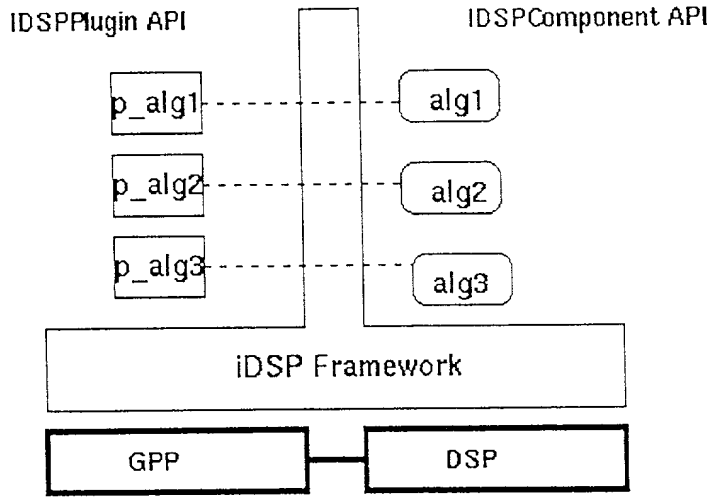

IDSPPlugin API
IDSPComponent API
p_alg1
p_alg2
p_alg3
alg1
alg2
alg3
iDSP Framework
GPP
DSP

Fig. 1a

iDSP: Plug, Play, QoS

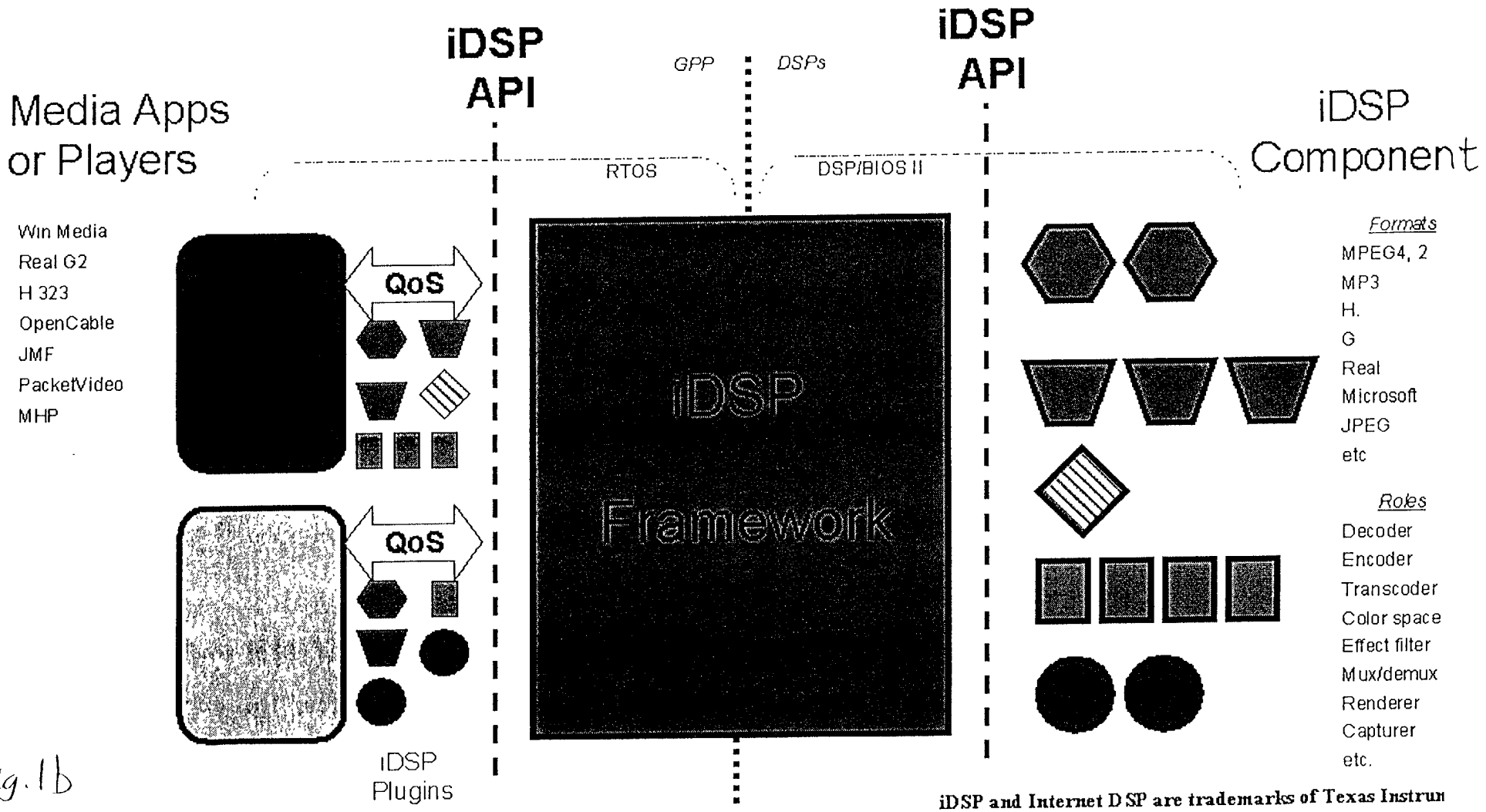


Fig. 1b

iDSP Algorithm Chaining

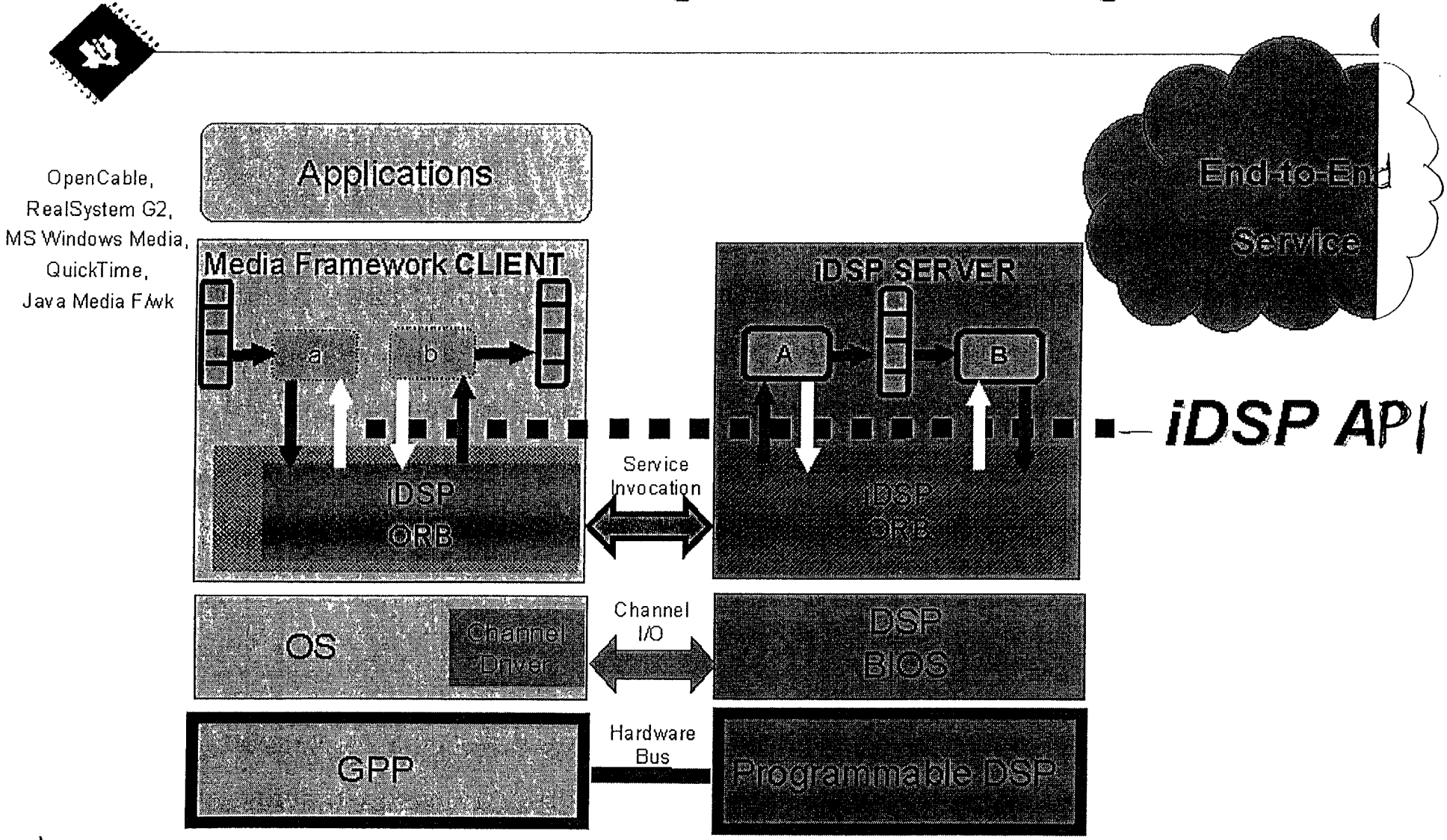


Fig. 1c

iDSP QoS Design

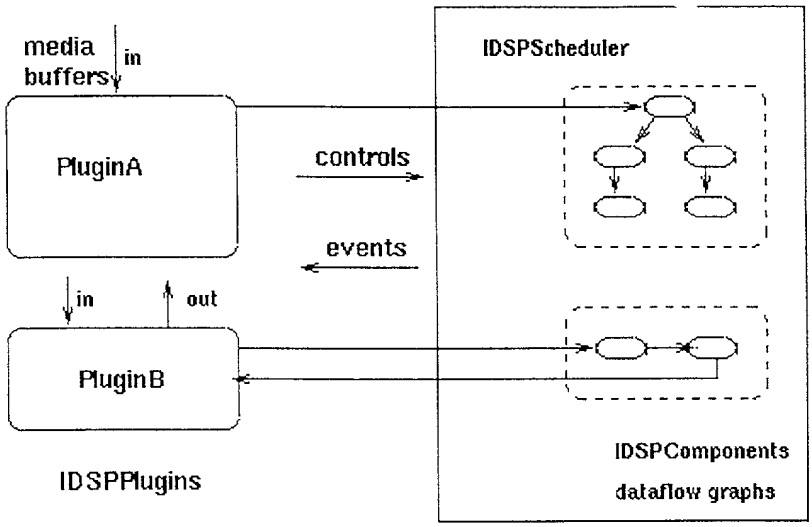


Fig 2a

IDSPQoSEvent types (received by IDSPPlugins)
* IDSPQoS_ALG_COMPLETED
* IDSPQoS_PRESENTATION_TIME_NOT_MET
* IDSPQoS_INSUFFICIENT_DATA
* IDSPQoS_INSUFFICIENT_CYCLES_AVAILABLE
* IDSPQoS_INSUFICIENT_MEMORY_AVAILABLE

IDSPQoSControl types:
* IDSPQoS_SET_RATE
* IDSPQoS_SET_QUALITY_LEVEL
* IDSPQoS_GET_STATS

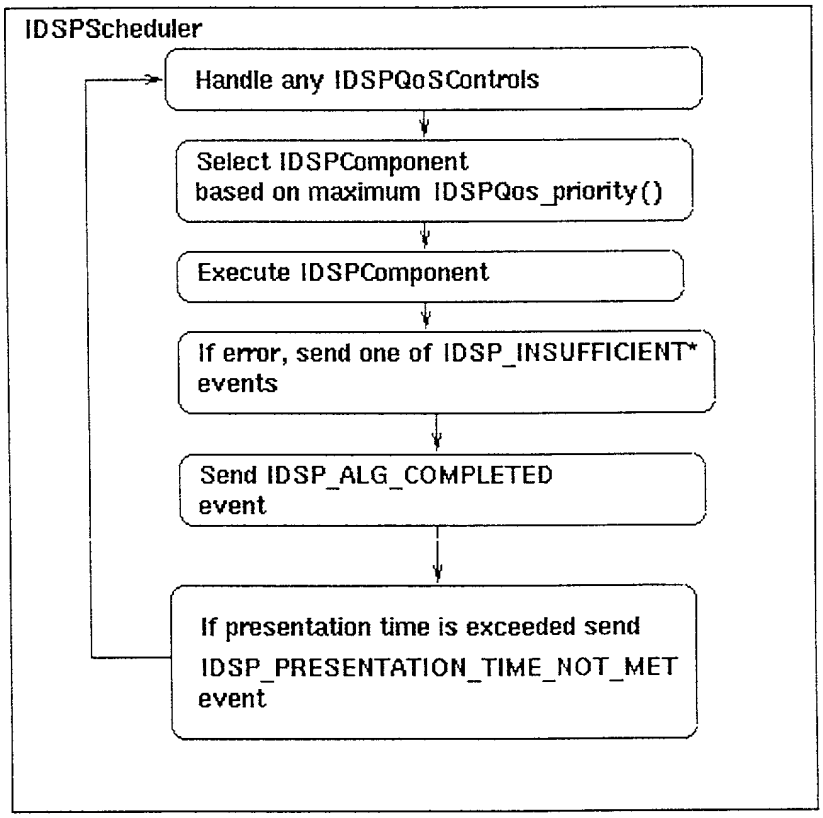


Fig 2b

- The IDSPScheduler provides QoS scheduling and event notification:
  - IDSPQoS_priority() is computed based on the time-criticality to meet presentation deadline. If the highest priority component cannot be rum, the IDSPScheduler analyzes the environent and sends an IDSPQoSEvent. The application can adjust the quality level or the rate.

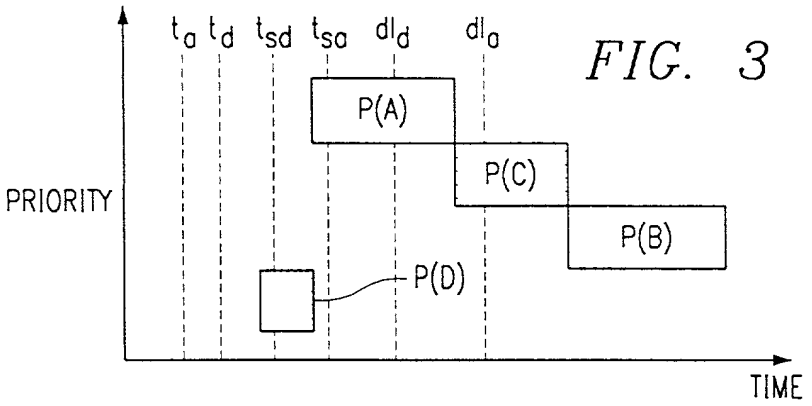


FIG. 3

t_{sa} = LAST POSSIBLE TIME FOR PROCESS A TO START AND STILL MAKES ITS DEADLINE

t_{sd} = LAST POSSIBLE TIME FOR PROCESS D TO START AND STILL MAKE ITS DEADLINE

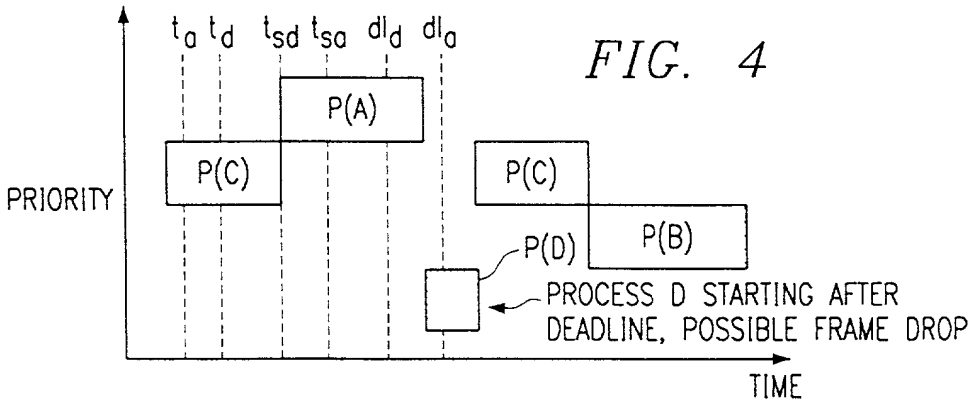


FIG. 4

t_{sa} = LAST POSSIBLE TIME FOR PROCESS A TO START AND STILL MAKES ITS DEADLINE

t_{sd} = LAST POSSIBLE TIME FOR PROCESS D TO START AND STILL MAKE ITS DEADLINE

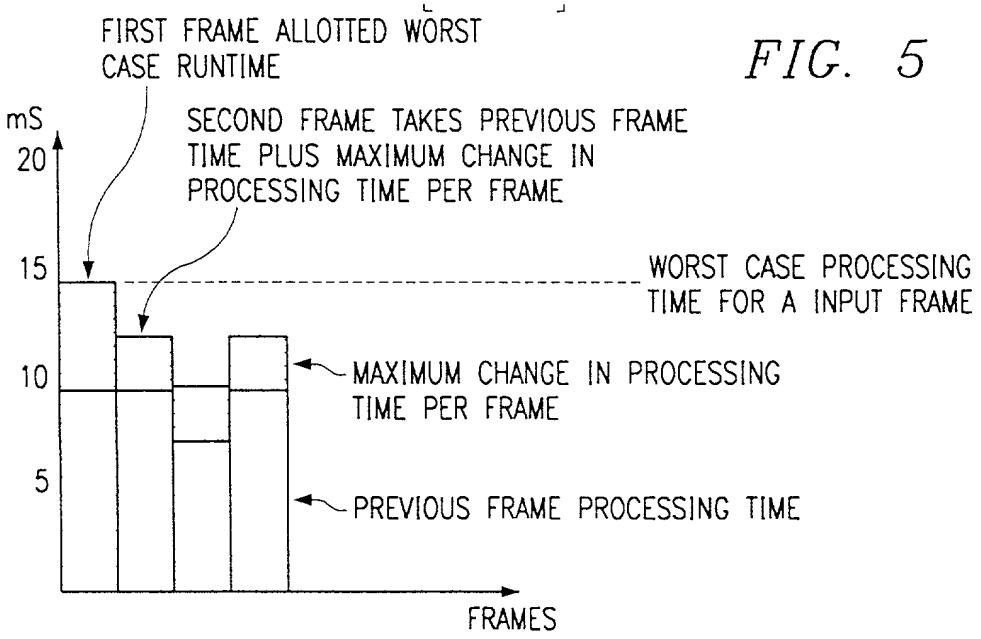


FIG. 5

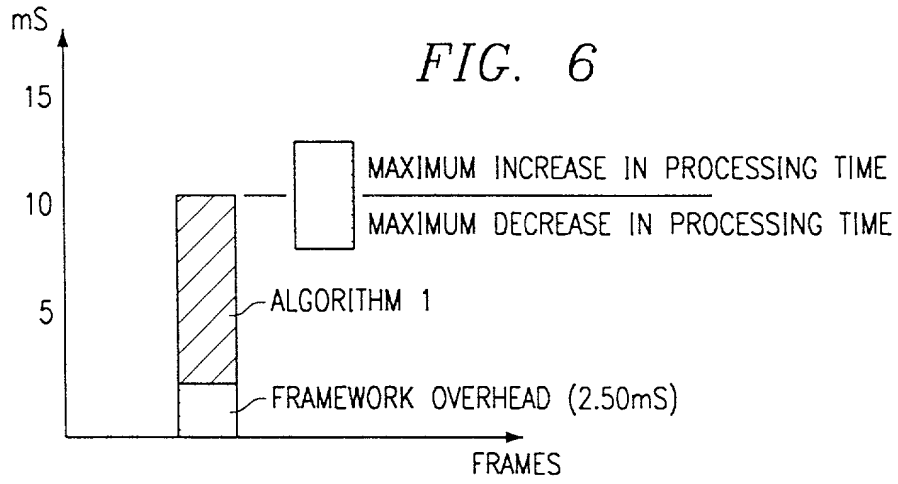


FIG. 6

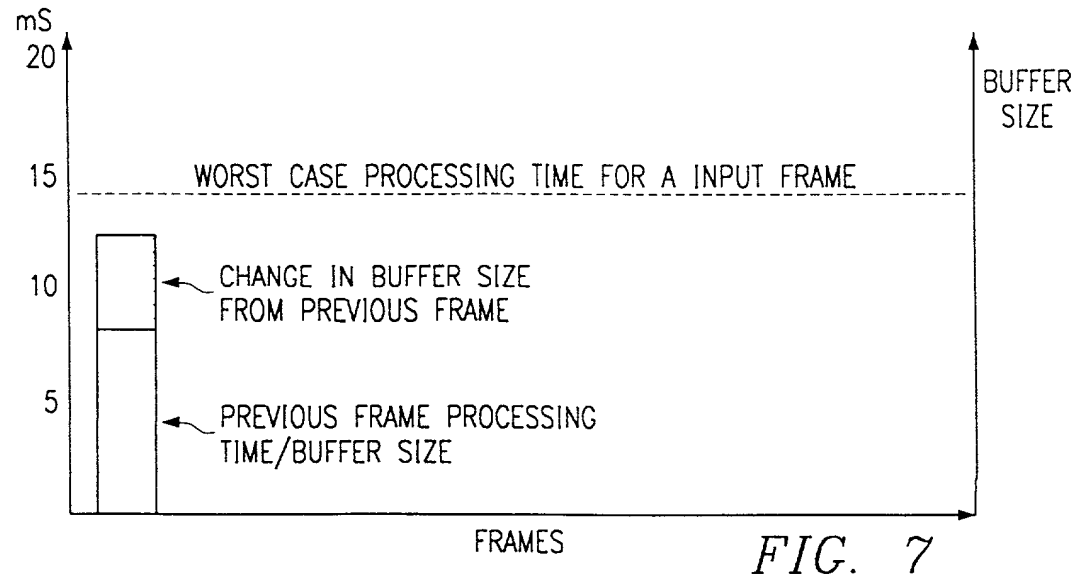


FIG. 7

FIG. 8

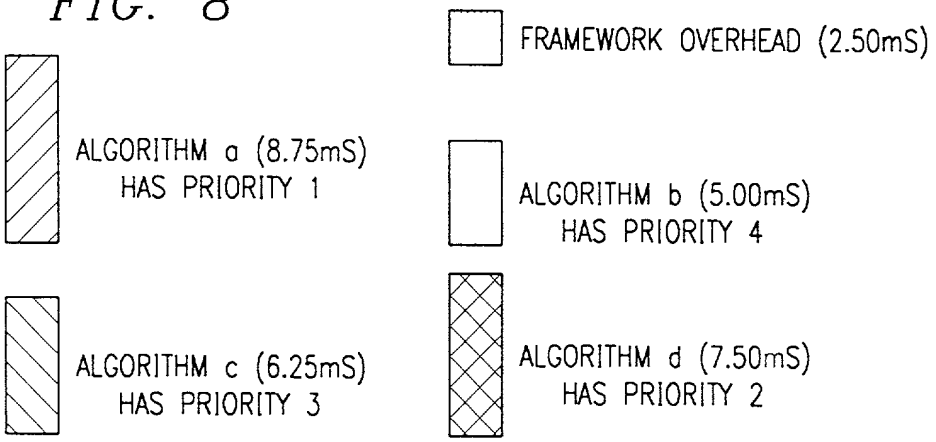
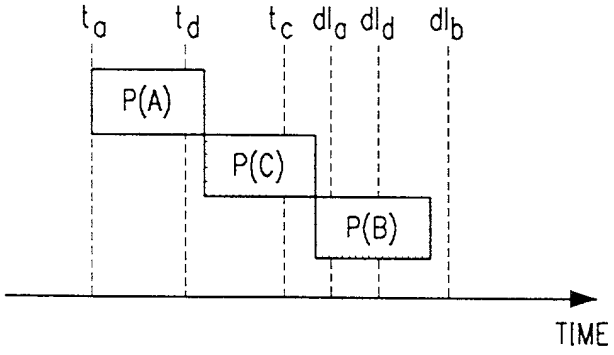


FIG. 9



t_i = TIME STAMP ARRIVAL OF EACH DATA FRAME FOR THE RESPECTIVE PROCESS

dl_i = DEADLINE FOR FINISHING PROCESSING OF EACH RECEIVED DATA FRAME

P() = PREDICTION OF PROCESSING TIME FOR EACH RECEIVED DATA FRAME

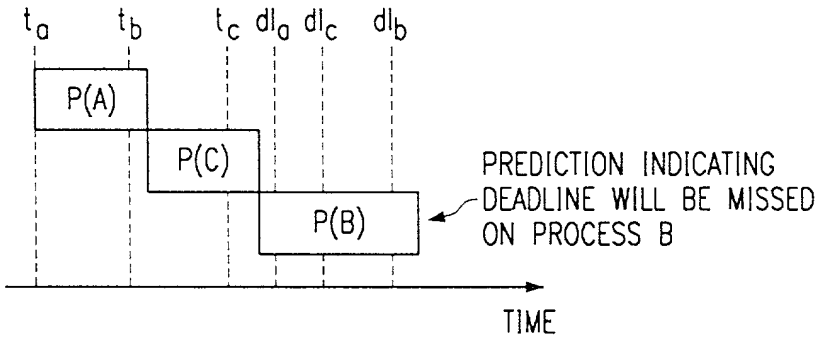
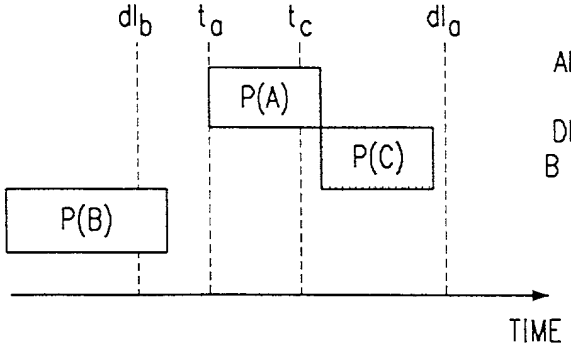


FIG. 10

t_i = TIME STAMP ARRIVAL OF EACH DATA FRAME FOR THE RESPECTIVE PROCESS

dl_i = DEADLINE FOR FINISHING PROCESSING OF EACH RECEIVED DATA FRAME

P() = PREDICTION OF PROCESSING TIME FOR EACH RECEIVED DATA FRAME



BOTH PROCESS A AND C ARE PREDICTED TO COMPLETE BEFORE THEIR RESPECTIVE DEADLINES MEANING PROCESS B MISSING ITS DEADLINE DOES NOT RIPPLE THROUGH THE SYSTEM (YET)

t_i = TIME STAMP ARRIVAL OF EACH DATA FRAME FOR THE RESPECTIVE PROCESS

dl_i = DEADLINE FOR FINISHING PROCESSING OF EACH RECEIVED DATA FRAME

P() = PREDICTION OF PROCESSING TIME FOR EACH RECEIVED DATA FRAME

FIG. 11

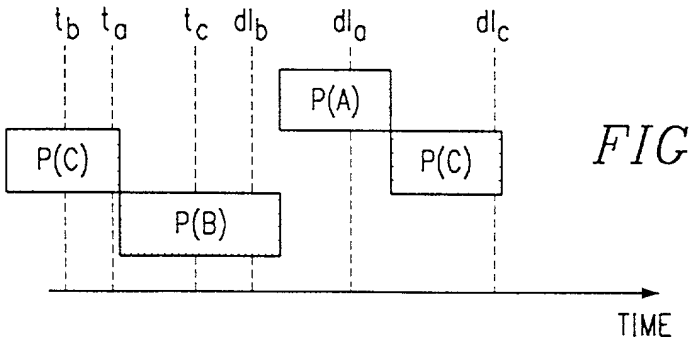


FIG. 12

t_i = TIME STAMP ARRIVAL OF EACH DATA FRAME FOR THE RESPECTIVE PROCESS

dl_i = DEADLINE FOR FINISHING PROCESSING OF EACH RECEIVED DATA FRAME

P() = PREDICTION OF PROCESSING TIME FOR EACH RECEIVED DATA FRAME

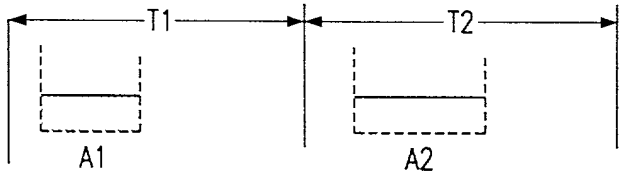


FIG. 13a

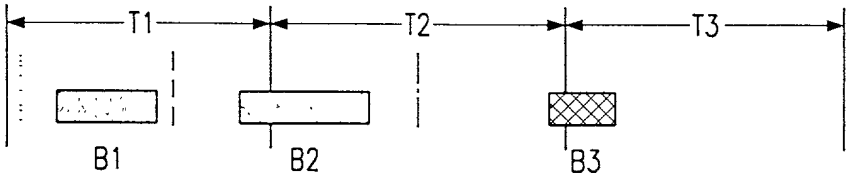


FIG. 13b

..... ARRIVAL OF BUFFER B1
- - - ARRIVAL OF BUFFER B2
—·— ARRIVAL OF BUFFER B3

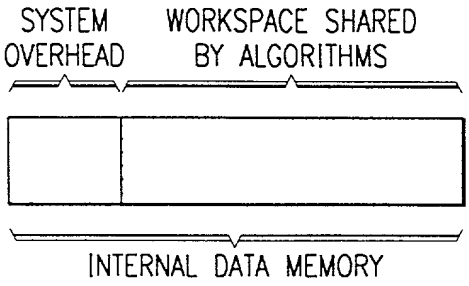


FIG. 14

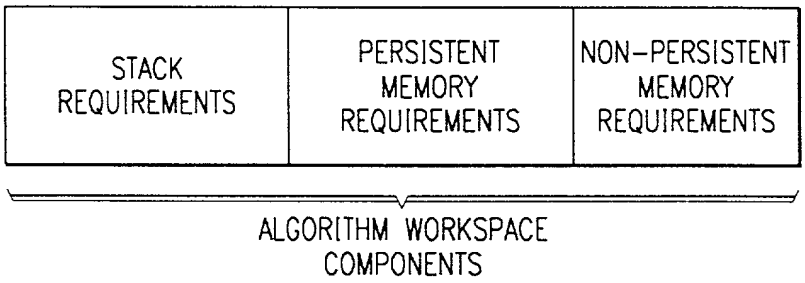


FIG. 15

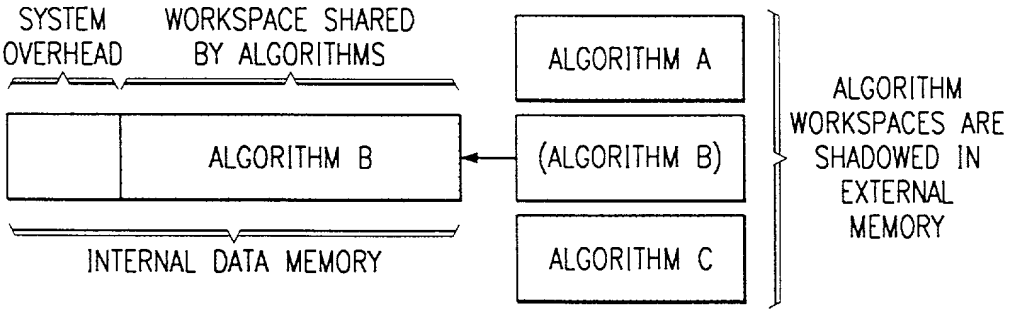


FIG. 16

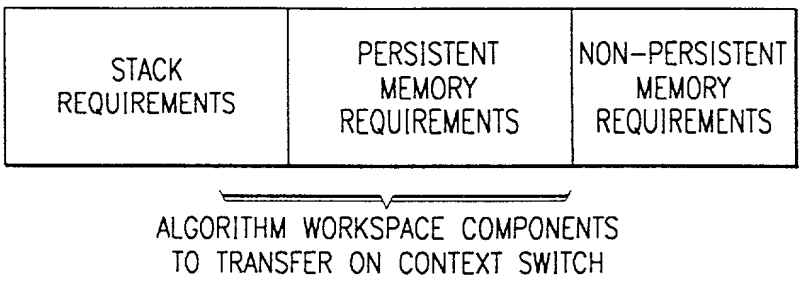


FIG. 17

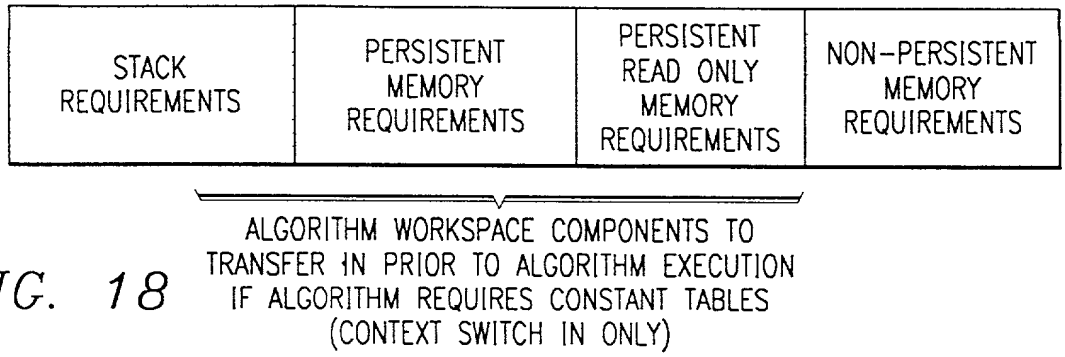


FIG. 18

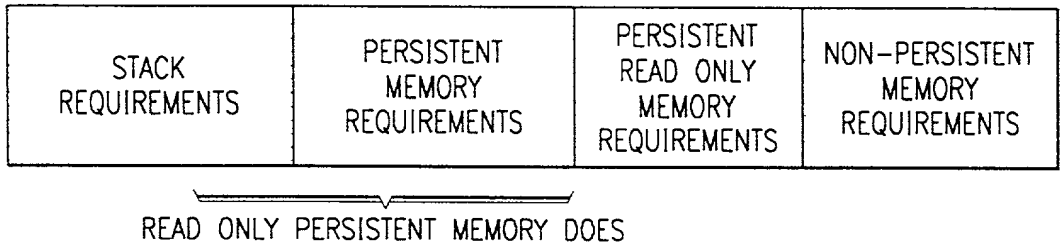


FIG. 19

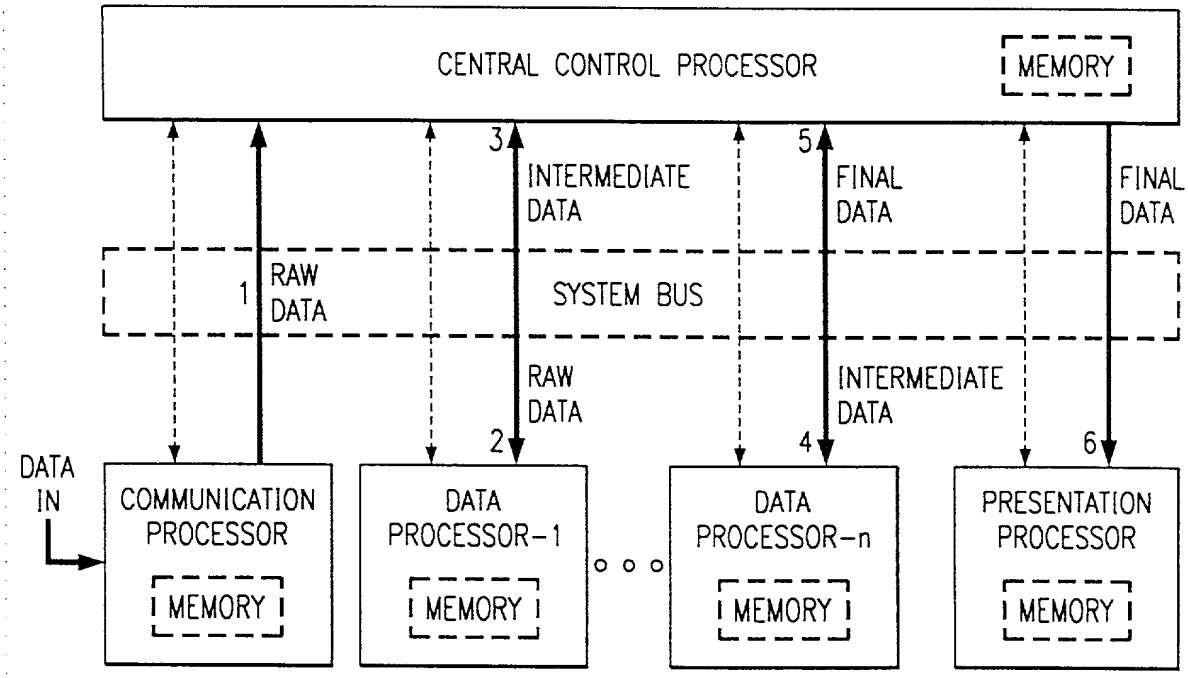
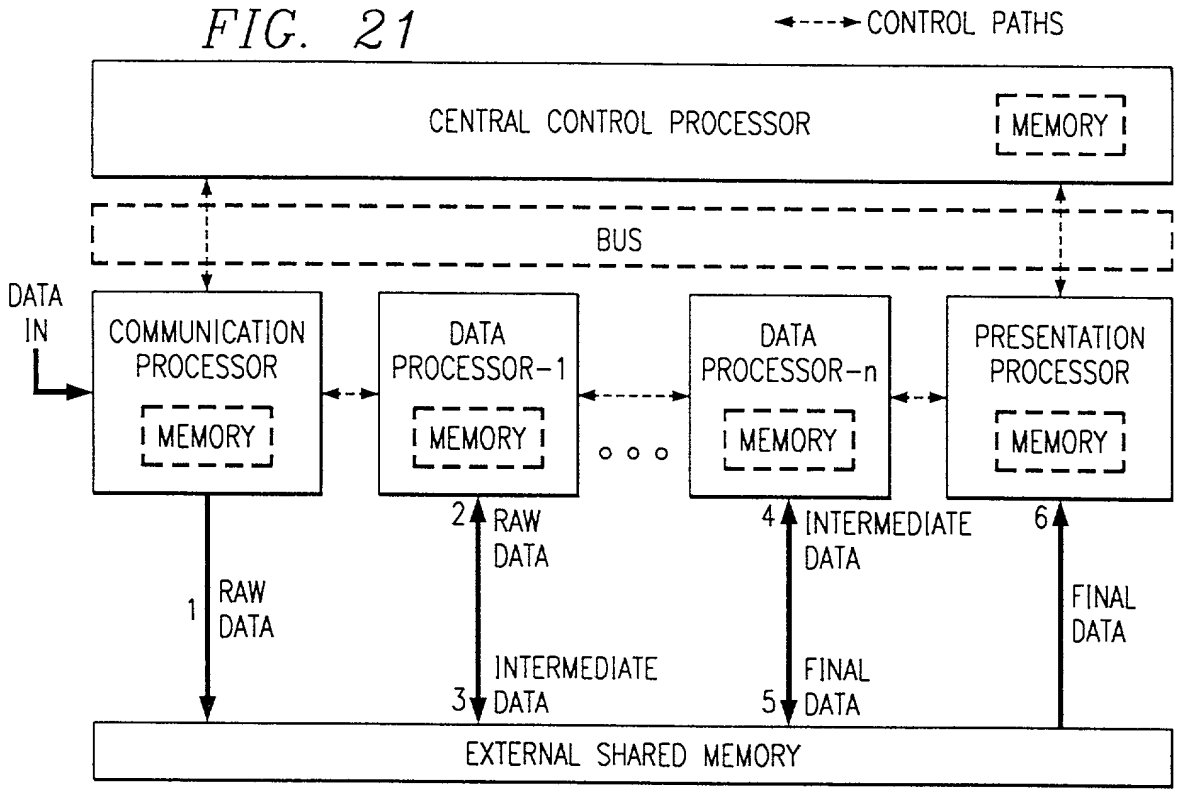


FIG. 20
(PRIOR ART)

FIG. 21


CONTROL PATHS
CENTRAL CONTROL PROCESSOR
MEMORY
BUS
DATA IN
COMMUNICATION PROCESSOR
MEMORY
DATA PROCESSOR-1
MEMORY
DATA PROCESSOR-n
MEMORY
PRESENTATION PROCESSOR
MEMORY
1 RAW DATA
2 RAW DATA
3 INTERMEDIATE DATA
4 INTERMEDIATE DATA
5 FINAL DATA
6 FINAL DATA
EXTERNAL SHARED MEMORY

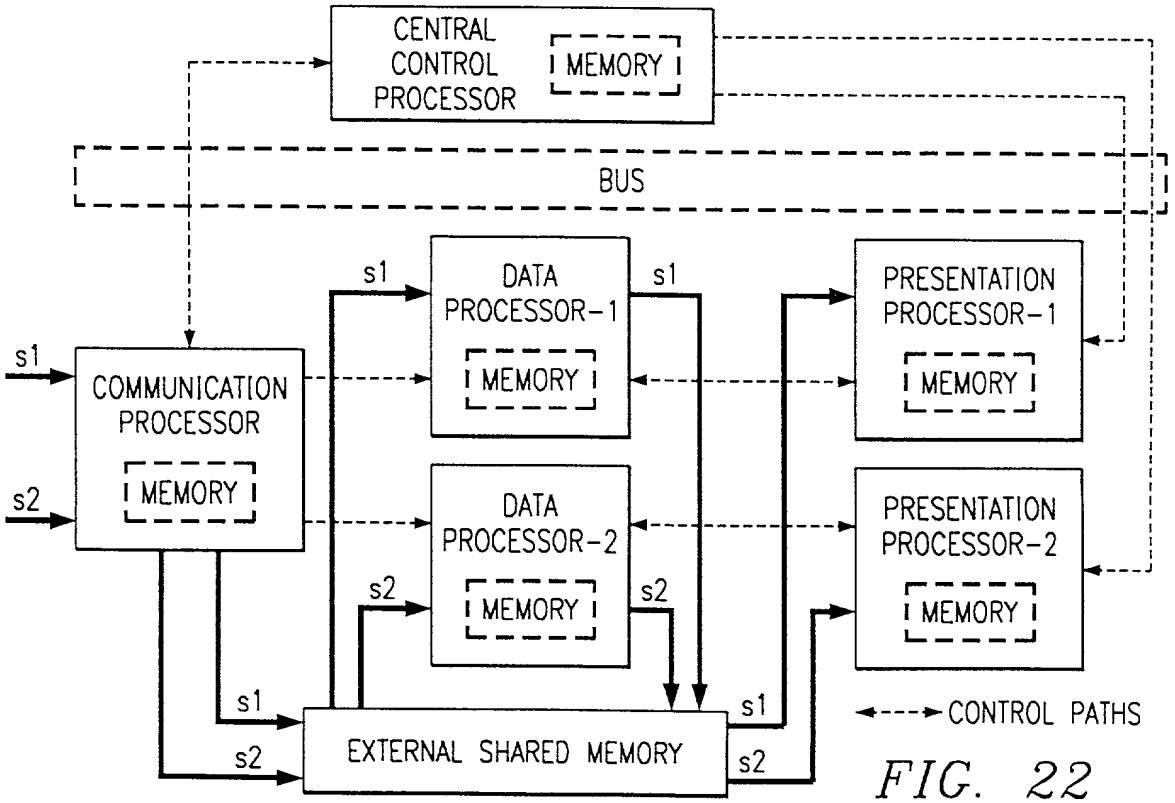
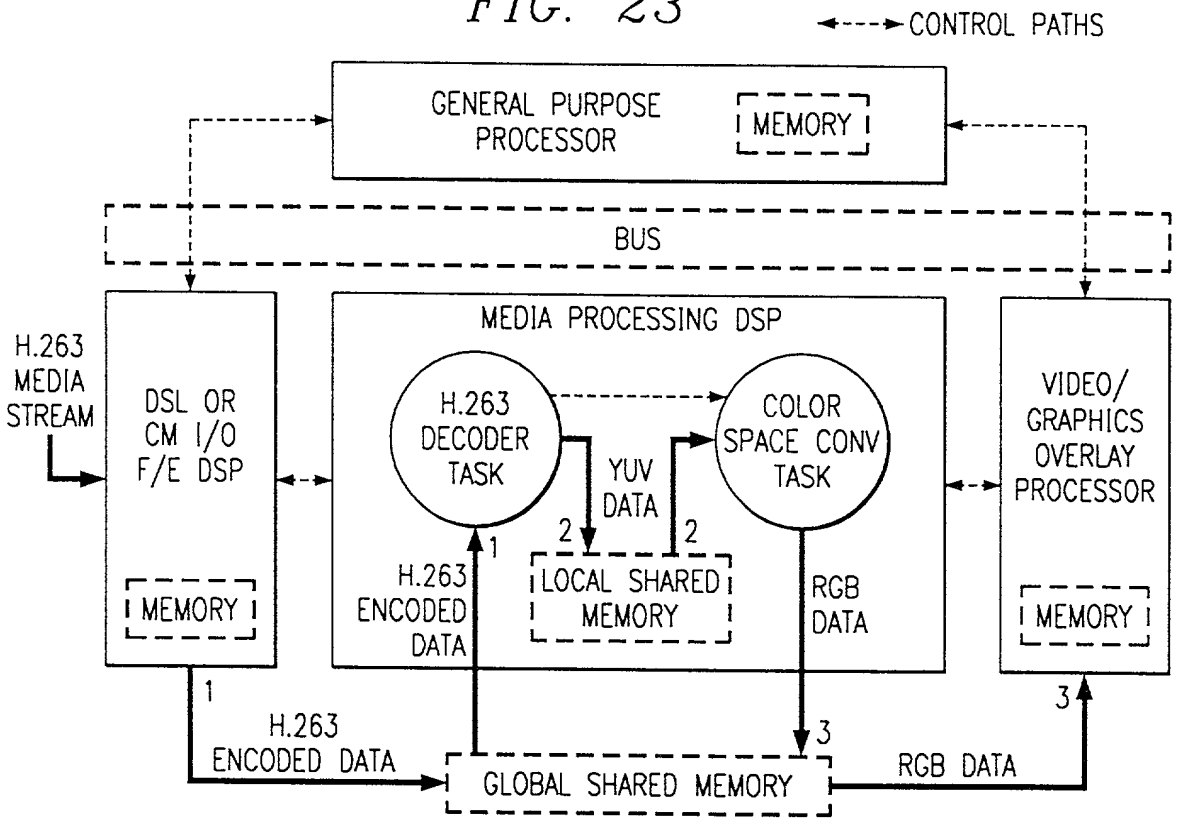

CENTRAL CONTROL PROCESSOR
MEMORY
BUS
s1
s2
COMMUNICATION PROCESSOR
MEMORY
DATA PROCESSOR-1
MEMORY
DATA PROCESSOR-2
MEMORY
PRESENTATION PROCESSOR-1
MEMORY
PRESENTATION PROCESSOR-2
MEMORY
EXTERNAL SHARED MEMORY
CONTROL PATHS

FIG. 22

FIG. 23


CONTROL PATHS
GENERAL PURPOSE PROCESSOR
MEMORY
BUS
H.263 MEDIA STREAM
DSL OR CM I/O F/E DSP
MEMORY
MEDIA PROCESSING DSP
H.263 DECODER TASK
YUV DATA
COLOR SPACE CONV TASK
H.263 ENCODED DATA
LOCAL SHARED MEMORY
RGB DATA
VIDEO/ GRAPHICS OVERLAY PROCESSOR
MEMORY
1
2
2
1
3
3
H.263 ENCODED DATA
GLOBAL SHARED MEMORY
RGB DATA

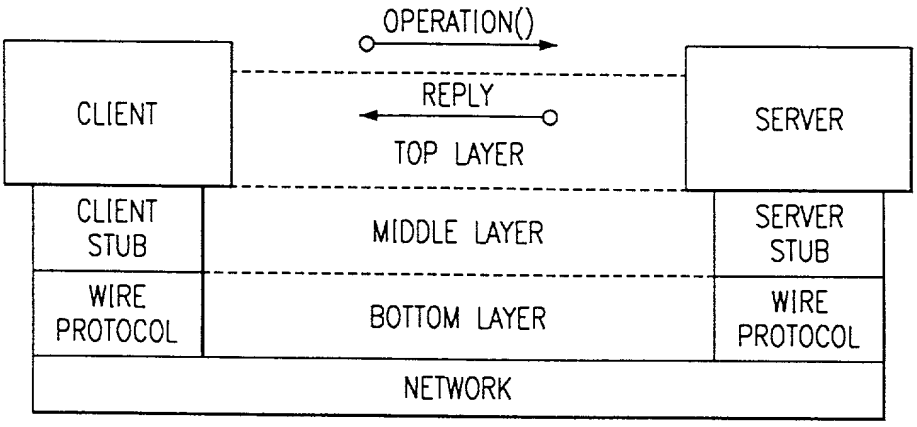

OPERATION()
REPLY
CLIENT
SERVER
TOP LAYER
CLIENT STUB
MIDDLE LAYER
SERVER STUB
WIRE PROTOCOL
BOTTOM LAYER
WIRE PROTOCOL
NETWORK

FIG. 24

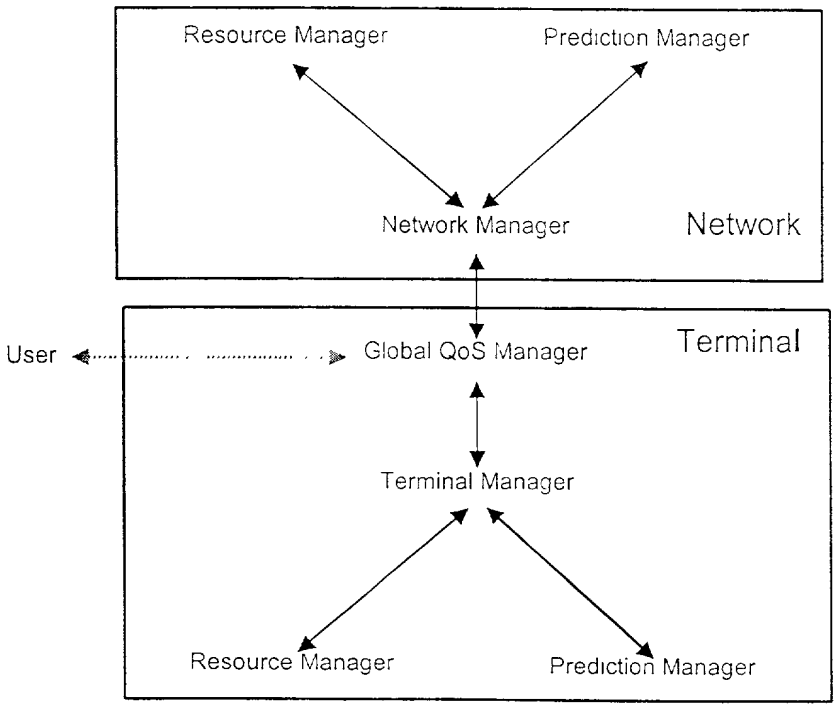


Figure 5: Simplified MPEG-21 Resource Management Framework

Figure25 shows a simplification of the proposed resource management framework, arrows denote control flow communication through APIs, not necessarily media flows. These control flows are governed by protocols. On the network side, these can coincide with

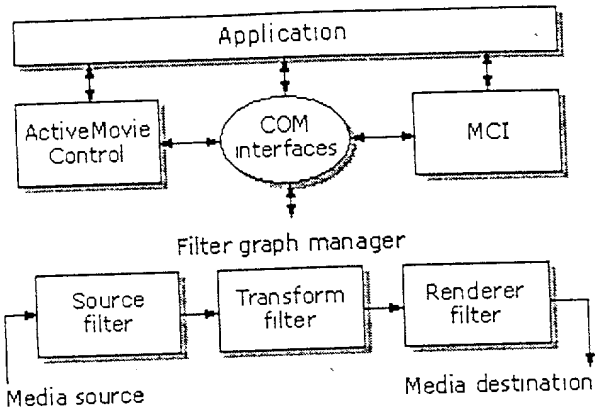


Fig 26

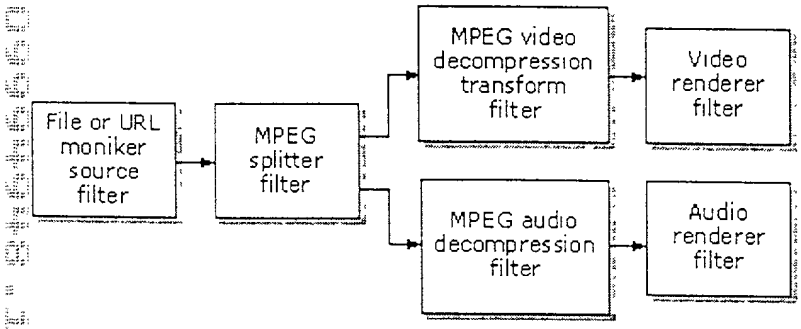


Fig 27

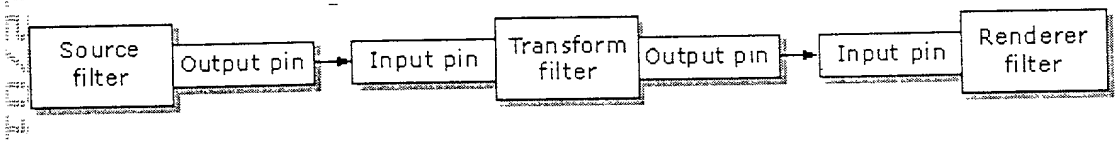


Fig 28